CHAPTER 6

Designing the Document Warehouse Architecture

Document warehouses, like their data warehouse brethren, have distinctive architectures different from online transaction processing (OLTP) systems. In this chapter, we will examine the components that make up the document warehouse. The key constituent components of a document warehouse are:

- Document sources
- Text processing servers
- Document storage options
- Metadata repository
- End user access and user profiles
- Support for data warehouse integration

In the next chapter, we will begin to delve into the processes that occur within the document warehouse, but our focus now is on the "what" and not the "how" of document warehousing.

The document sources provide the raw material for the warehouse. Text processing servers analyze the text and extract salient information. Thanks to the large, high-performance demands of data warehouses and online transaction processing systems, we have a number of storage architecture options that we

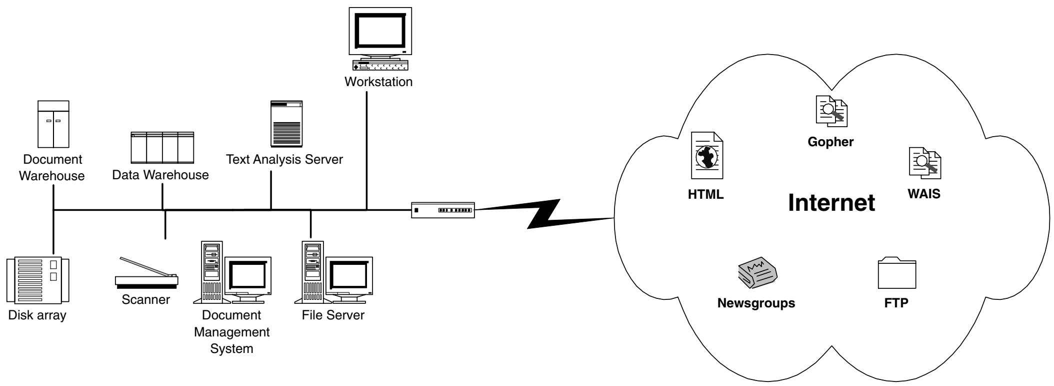


Figure 6.1 The architecture of a document warehouse.

will examine in detail. We will also look into the logical structure of document warehouses, including ways of managing metadata. Finally, the components that will ultimately provide end user access to the document warehouse will be presented. Figure 6.1 shows a general example of a document warehouse environment.

Document Sources

Document warehouses are populated from multiple sources, including:

- File servers
- Document management systems
- Internet resources

Clearly, there are overlaps in these areas. Document management systems and intranets both use file servers to manage files. However, based on their differences at their highest levels, we will consider them distinct.

When we look at each document source, we will ask several questions. What types of documents are available? How are the files accessed? What limitations of these document sources will affect the document warehouse? What features can be exploited for the benefit of the warehouse? The answer to these questions will affect how we design document retrieval programs and how we extract metadata. We will start with the simplest document source, the local file system. Figure 6.2 shows the key focal area of this section.

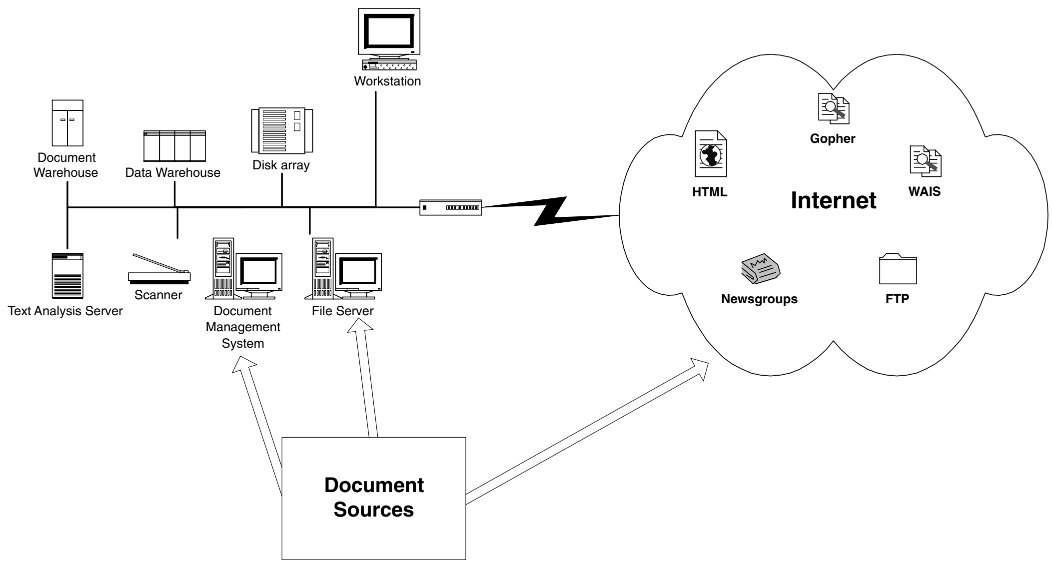


Figure 6.2 Document sources in the document warehouse environment.

File Servers

File servers are ubiquitous in the networked organization. In this section, we will briefly discuss the functional features of file servers and then various configurations to support a range of document-processing requirements.

File Server Functions

The primary purpose of these servers is to provide organized storage for a broad range of documents with as few restrictions as possible. For example, in UNIX file systems, files are treated simply as a stream of bytes. This simple assumption makes it very easy to store a variety of types of files, from text and spreadsheets to graphics and computer-assisted design (CAD) documents. With so few assumptions about the structure of a file, operating systems are able to provide only a few basic operations such as:

- Copying
- Deleting
- Versioning
- Purging
- Linking

Copying and deleting are common to all operating systems. The ability to store multiple versions of a single file is available in some operating systems, such as OpenVMS, along with the ability to remove old versions by using purge commands. Linking is the ability to create pointers to a file to make a virtual copy, which allows one to store a single copy of a file yet have it appear in multiple directories. This is another useful feature, but it is limited to only some operating systems, such as UNIX.

File servers are the wide-open ranges of document warehousing. The vast majority of an organization's documents are stored on file servers that are not themselves readily available from the Internet, or on specialized applications such as document management systems. The key benefit here is that documents are easy to retrieve for inclusion in the warehouse by using operating-system-level commands. Since most operating systems provide scripting languages and support even more powerful file manipulation tools, such as Perl and Python, extracting and loading documents from a file system is trivial. (Perl and Python are both full-blown programming languages useful for many different programming tasks, but their support for file manipulation makes them ideal tools for the extraction, transformation, and loading phase of document warehousing.)

Like a double-edged sword, the reasons that file systems are so easy to work with are also the reasons that they present difficulties for document warehousing. First, since most operating systems lend themselves to naming conventions, but do not enforce them, we can run into problems if we depend on file names to distinguish file types. As long as everyone agrees to save all word-processing files with a .doc extension, then there is no problem finding such files. More importantly, regularly produced documents may be named according to a complex scheme. For example, a consulting company may have a directory for projects with subdirectories for each individual project. Within each subdirectory, each file may be named according to its type—for example, status report, invoice, or proposal. Extracting all project status reports from the file system is then just a matter of searching each subdirectory, assuming that the naming convention has been followed. Without enforcement of these conventions, though, we could easily miss documents that should be included or include those that should be excluded.

Another limitation of file systems is that they do not store rich metadata about documents. All operating systems store some information about a file, such as creation date and time, last access time, and attributes such as archive flags. They do not include higher-level metadata such as author (although some have the username of the file owner, this is not the same), keywords, and description of purpose.

So far we have addressed the logical functions of file systems, but now it is time to turn our attention to the storage configuration options of these systems.

Storage Configurations

Broadly speaking, storage devices are generally configured in three different ways:

- Disk off server
- Network attached server
- Storage area network

Each configuration offers different benefits, and which on you choose depends on the particular needs of your document warehouse. While we are discussing storage configurations with regard to data sources in this section, the same points apply to storage for the document warehouse itself.

Disk off Server

The disk off server is the simplest configuration. In this scenario, disks are physically attached to a server using a high-speed bus such as the 160 MB/sec Ultra3 SCSI. The server itself is a node on a subnetwork shared with other servers and client machines as depicted in Figure 6.3.

With this configuration, storage is managed as part with the server thus minimizing additional administration overhead. The drawback is that the storage device is dependent upon the server, which may support other tasks as well. Connecting a storage device directly to the network, as with network attached servers, can eliminate that problem.

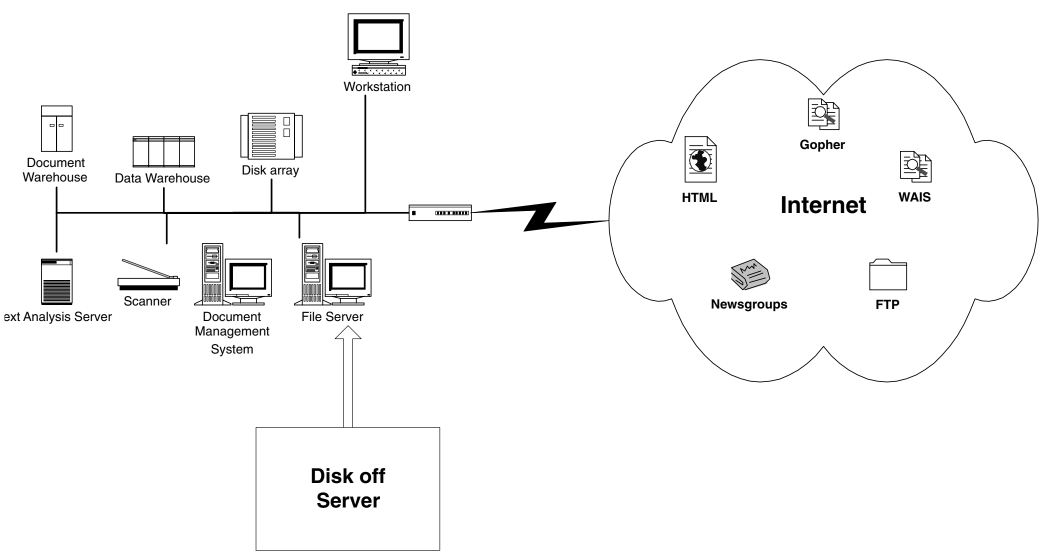
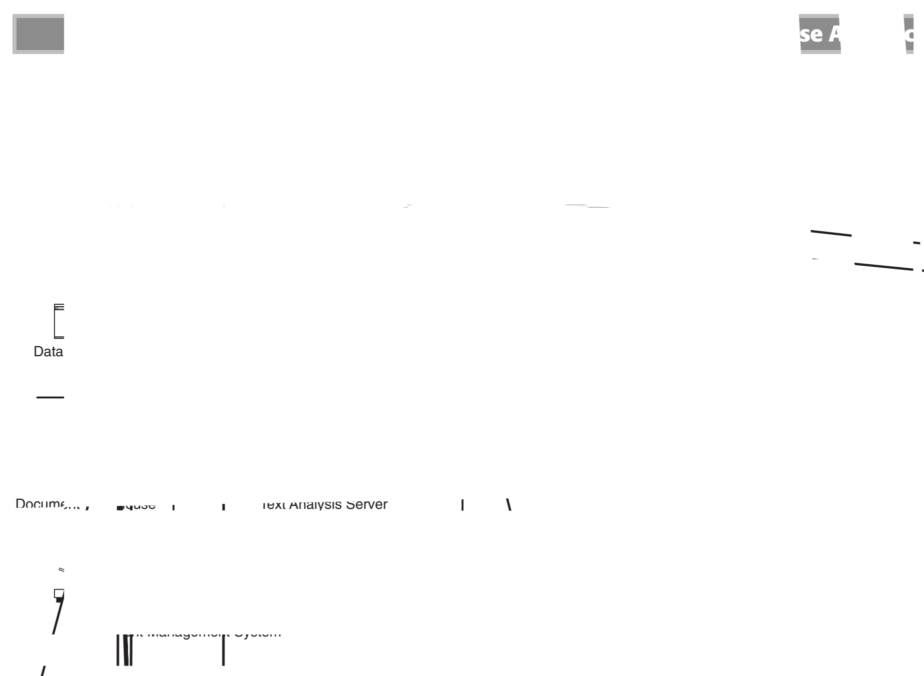


Figure 6.3 Disk off server configuration depends upon the server's network connection.

Network Attached Servers

Network attached servers (NASs) consist of disk arrays and other storage devices that connect directly to a network through their own network interface. NAS devices generally use 10-Mbps or 100-Mbps Ethernet or other high-speed network


Data
Text Analysis Server

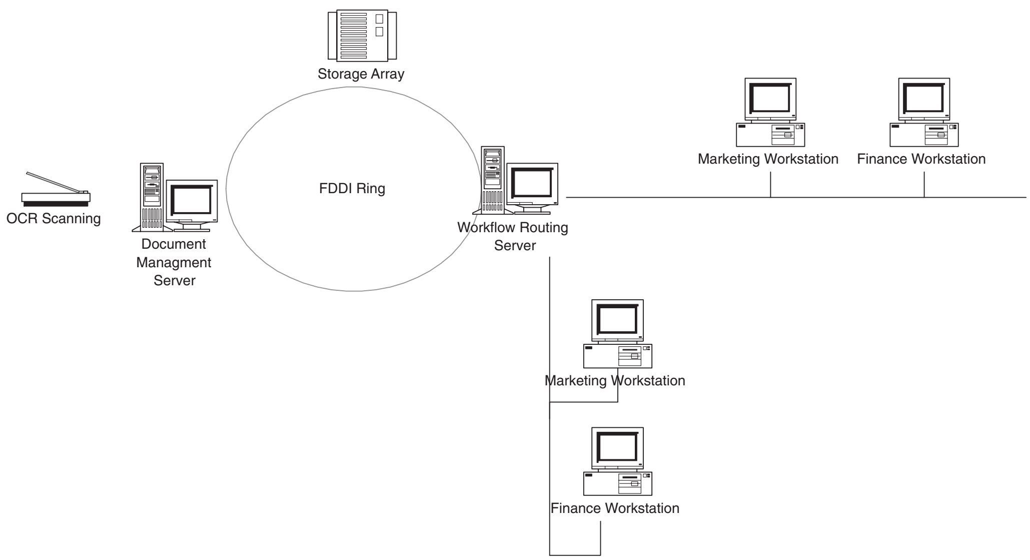


Figure 6.6 A document management system can include scanning, storage, and workflow routing functions.

For our purposes, we are primarily concerned with extracting documents and related metadata from document management systems for long-term inclusion in the document warehouse. The rising popularity of document management systems, which often operate at a departmental and not enterprise-wide level, has led to a proliferation of a number of different document management systems that need to interoperate. The document management industry has responded to the need by developing standards and programming interfaces to these systems. At the same time as document management vendors were wrestling with the interoperability problem, Web developers we addressing similar problems with content management on HTTP servers. The Internet Engineering Task Force (IETF) has developed a distributed authoring standard that is proving useful in the document management arena as well. We will now briefly turn our attention three of these standards:

- Open Document Management API
- Document Management Alliance
- WebDAV

Each standard provides a distinct approach to the problem of document management. The first two grew out of the document management industry, while WebDAV is an Internet standard that provides many of the benefits of the others while offering a widely adopted standard.

Open Document Management API

The Open Document Management API (ODMA) standard is a platform-independent application programming interface that provides a client application interface to document management systems. Widely used desktop applications support the ODMA—including Microsoft Word, Corel WordPerfect, and Visio's diagramming product. The goals of ODMA were to minimize the burden on application developers dealing with document management systems and to provide platform independence. To that end, ODMA addresses:

- Unique document identifiers
- Error handling
- Connections and a connection manager
- Document format names
- File system dialogs
- Character sets
- Application interfaces
- Document management system querying

With our focus on extracting documents, the query support ODMA provides is especially important. First of all, ODMA provides a number of functions to manipulate documents within the document management system, including:

- ODMGetDocInfo: This function returns information about the document from the document management system.
- ODMOpenDoc: This function makes a specified document available to the application.
- ODMQueryCapability: This function is used by clients to determine if a document management system provides support for a particular ODMA function.
- ODMQueryExecute: This function has a document management system parse a string representation of a query and return a query ID that is used by ODMQueryGetResults to retrieve document identifiers.
- ODMQueryInterface: This function is used to get a COM interface from an ODMA provider.

Some of these functions, such as the ODMGetDocInfo and Select queries can use specific attributes supported by ODMA. Some of the most important are:

- ODM_AUTHOR: Author of the document
- ODM_CONTENTFORMAT: Format name string indicating the contents using either MIME Content Type or Windows file type or extensions

- ODM_CREATEDBY: Username of person who created the document
- ODM_CREATEDDATE: The date and time the document was created
- ODM_DOCVERSION: The document version string
- ODM_LOCATION: The logical location (e.g., folder)
- ODM_KEYWORDS: A comma-separated list of keywords assigned to a document
- ODM_NAME: A descriptive name of the document but not the file name
- ODM_OWNER: The document owner
- ODM_SUBJECT: A string describing the contents of the document
- ODM_TITLETEXT: A short descriptive title of the document, possibly compiled from a document's profile
- ODM_TYPE: The type of the document, such as a memo, contract, status report, and so on
- ODM_URL: The Universal Resource Locator of a document

These attributes are used in Select statements that are at the heart of queries. For example, to find the author and location of all contracts, we could use the following statement:

```
Select
     ODM_DOCID, ODM_AUTHOR, ODM_LOCATION
Where     ODM_TYPE = 'contract'
```

For extracting documents from a document management system, ODMA provides many of the features needed to program extraction routines using a portable interface. While ODMA focuses on the function and COM level interaction between client applications and document management systems, another standard, the Document Management Alliance standard, addresses broader architectural issues.

Document Management Alliance

A consortium of over fifty commercial users, vendors, integrators, and government organizations formed a working group, the Document Management Alliance (DMA), to address interoperability in document management systems under the auspices of the Association for Information and Image Management (AIIM). Some of the key features of the DMA standard are:

- Automatic location of document repositories
- A common mapping for attributes across document management systems
- Support for document versioning

- Support for folders
- Support for Web browsers
- Support for multiple renditions of a document
- Automatic discovery of document classes and properties
- Support for the UNICODE character set

The DMA standard works with the ODMA function set, but also provides additional features to hide implementation-specific characteristics of the document management system. The resulting goal of the DMA standard is to provide true many-to-many interoperability between document management systems. True many-to-many interoperability comes at a cost, though, because the DMA standard is more complex than the ODMA. A third alternative, which again can operate with or independently of DMA, is the WebDAV standard.

WWW Distributed Authoring and Versioning

While ODMA may be too low level a standard for some, and the DMA standard may be too complex for others, the WWW Distributed Authoring and Versioning (WebDAV) standard provides something of a middle ground. WebDAV grew out of the needs of Web developers to better manage Web sites with features such as remote editing and loading and saving of documents and other media types. Some of the key features of WebDAV, defined in Internet Engineering Task Force (IETF) documents RFC 2291 and 2518, include:

- Document properties
- Typed connections called links
- Locking
- Reservations of documents
- Partial writes
- Name space manipulation
- Collections
- Versioning
- Security
- Internationalization

Furthermore, these features operate across replicated distributed servers. Since many of the features important for Web administration are also found in document management systems, it is not surprising that WebDAV has been adopted for some applications. WebDAV is the protocol behind Microsoft Office 2000

Web Folders and the interface to the Microsoft Exchange 2000 Web Storage System. WebDAV is also supported in the Apache Web Server, the Perl programming language through the PerlDAV module, and Adobe's GoLive product.

DAV Searching and Locating (DASL) adds improved functionality to WebDAV. The purpose of this standard is to extend the search capabilities of DAV to include:

- Finding resources of a particular kind
- Finding resources of a particular language
- Using content and property searches
- Word stemming
- Word proximity searches
- Query by example

DASL will also work with taxonomies and thus brings key text mining operations within the realm of a widely implemented Internet standard.

Through WebDAV and DASL, we can see how document management systems and the Internet may converge more tightly and blur the lines of distinction between what lies within and outside of a document management system. For the time being though, we will continue to treat them as distinct entities, since standards such as ODMA and DMA are implemented in existing systems, and the new DASL standard is still emerging.

Internet Resources

As we saw in Chapter 4, the Internet is a rich source of documents for text mining and the document warehouse. One way to look at the logical structure of the Internet is through the protocols it supports, and here we will consider four protocols as indicative of important architectural elements for document warehousing. The protocols we will consider are:

- HTTP
- FTP
- Gopher
- WAIS

HTTP is the most popular and is the base protocol for the World Wide Web. XML Linking and Pointing, the newest of the protocols, is quickly addressing some of the shortcomings of HTTP, and is commonly used for transferring files, while Gopher and WAIS have, to some degree, been eclipsed by HTTP. Next, we will briefly discuss the role played by these protocols, and in the next chapter we will look in greater detail at how to exploit these tools when searching for document warehouse sources.

HTTP

The Hypertext Transfer Protocol (HTTP) protocol provides the means to link documents and other resources. When we think of text mining, we usually mean text mining in a hypertext environment using either HTML or XML Pointing and Linking. Both build upon HTTP. Along with Universal Resource Locators (URLs), it provides the building blocks for retrieving documents from the Internet, intranets, and extranets.

FTP

The File Transfer Protocol (FTP) provides Internet-based access to hierarchically organized file systems or parts of file systems. One of the key benefits for document-loading processes is that document-loading programs built on FTP become portable across file systems. Of course some file system idiosyncrasies may need to be attended to, but for the most part the core command set is the same across server types.

Gopher and WAIS

Gopher provides a hierarchical organization of text on the Internet, while the Wide Area Information Servers (WAIS) support searching for text on topically organized WAIS servers. Neither protocol is as widely used as HTTP or FTP but some early sites still support them.

From Document Sources to Text Analysis

Finding documents is the first step in the document warehousing process, and our main sources are file systems, document management systems, and the Internet. Each type of source has its own distinct set of architectural features and processing issues that we must bear in mind when designing the document warehouse. For example, file systems have relatively straightforward utilities and commands for manipulating documents, the Internet provides several protocols, reflecting different organizational structures, and document management systems may require the use of a proprietary API or one of a few standard interfaces to their document repository. In any case, once the means of extracting the documents have been defined, we will need to get them to servers for text analysis.

Text Processing Servers

Text processing servers are responsible for getting documents, analyzing them, and distributing them as needed. We will first examine how documents can be retrieved using agents and crawlers and then look into how to configure text analysis servers to appropriately balance the workload in a document warehouse. Figure 6.7 depicts text processing servers in the middle of the document warehouse, reflecting their central importance to text mining operations.

Using Crawlers and Agents to Retrieve Documents

Agents and crawlers are programs designed to work in a networked environment. Crawlers are programs that run on a single server and retrieve resources—such as Web pages—and use information within that resource to find other resources. In document warehousing, an agent is a program that may run on different servers in a network environment to carry out a specific task, sometimes communicating with other agents.

Crawling the Web

Anyone who has used a search engine has benefited from the use of crawlers. These programs are fairly straightforward. An indexing crawler is given a start-

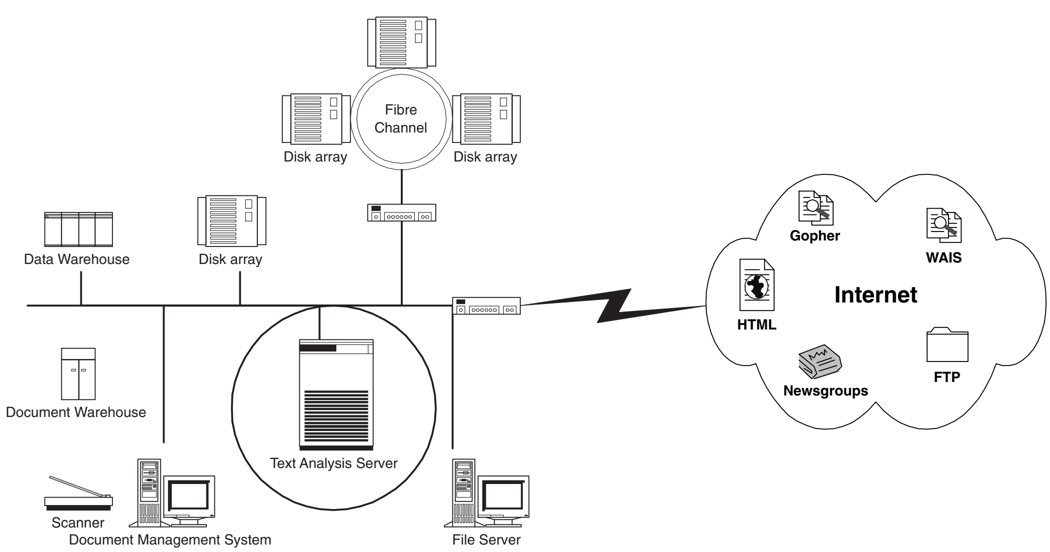


Figure 6.7 Text processing servers are the heart of the document warehouse.

ing point, usually in the form of a URL. The crawler downloads the document at the specified URL and then extracts all of the hypertext links to other resources from that page. The crawler then repeats the fetch and extract processes, collecting documents as it goes.

Not all crawlers are strictly for searching and indexing. Crawlers exist for extracting specific pieces of information, such as the cost of a book or the times of airline flights. These programs are frequently called agents, but here we will reserve that term for a more complex type of system. Rather than extract hyperlinks, these programs fetch a Web and extract particular pieces of information, such as product descriptions and cost or scheduled departure and arrival times of flights. The principles and functions of search and index crawlers remain the same. Custom crawlers can be designed for document warehouses and other decision support environments with specific needs. For example, an energy industry analyst may want to gather crude oil price projections from a number of different sources for comparison. In this case, as with crawlers designed for comparison shopping over the Internet, a number of specific sites are visited, and targeted information is extracted. Exactly how the information is extracted is dependent upon the document retrieved. XML documents can be easily parsed, while free-form text must be more carefully analyzed to extract relevant information with custom routines.

Crawlers are uncomplicated but powerful programs. Although each step is simple, when done over and over in a richly interlinked environment, the collective results can be an expansive search of the hyperlinked environment. Another way to effectively search a distributed network of documents is with agents.

Agents in Action

Agents differ from crawlers in several significant ways. First, agents are directed to accomplish a specific task, such as retrieving information about a particular topic. Search crawlers, on the other hand, move from document to document through links without regard for a reader's topic of interest. Second, agents can interact with other agents to share information, while crawlers work independently. A key benefit here is that an agent can acquire information, such as a list of interesting sites, from another agent and thus avoid having to compile the list itself. Third, because agents interact, more complex architectures can be developed on the basis of specialized agents. For example, some multiagent systems found in information retrieval use producer information for other agents, consume information and deliver it to end users, and organize tasks. Yet another distinction is that agents can persist over time, continuing to look for information. Agents with long lifetimes, such as weeks or months, become "Smart Browsers," continually browsing the network looking for topics of interest, perhaps revisiting sites and constantly sending back new information (Marsh and Masrour

1997). While a number of different approaches have been developed for implementing multiagent information retrieval systems, they share a common general architecture, which we will discuss next.

Multiagent Architecture

Both commercial systems such as MCC's InfoSleuth (www.mcc.com/projects/infosleuth), and research projects, such as Marsh's ACORN (Marsh and Masrour 1997) and Decker et al.'s MACRON (Decker, et al. 1995), share enough common architectural features that we can generalize a basic multiagent architecture for information retrieval. It should be noted that these applications are designed for discovering and retrieving information but not necessarily for a document warehouse or for text mining purposes. Figure 6.8 shows the basic multiagent architecture.

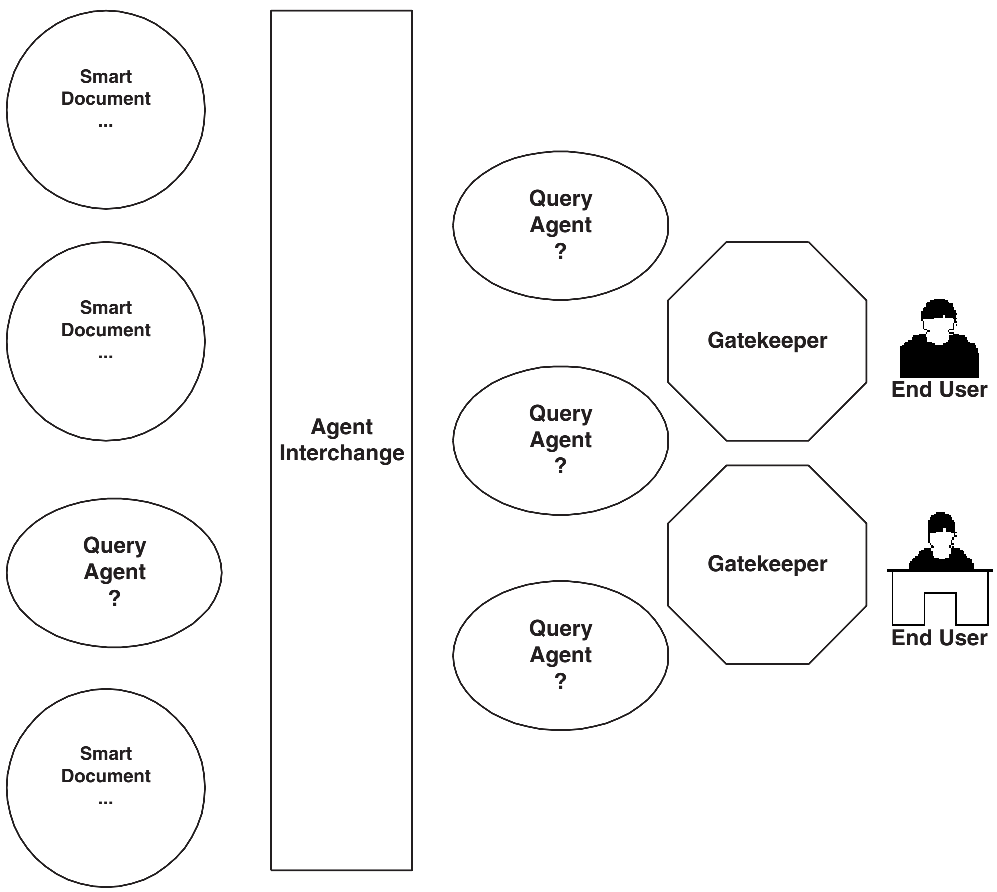


Figure 6.8 A generalized multiagent architecture for information retrieval.

The basic components of multiagent systems include:

- Consumer or query agent
- Producer or smart document agent
- Agent interchange
- Gatekeeper agent

Consumer or query agents represent a user's area of interest, such as the rate of production of crude oil and natural gas. The representation could be a simple list of keywords, a set of terms from a taxonomy, or a query in a more complex language such as the Knowledge Query Manipulation Language (KQML). The goal of consumer agents is to find producers representing documents that match the consumer's interest. Frequently, a consumer will start with a list of potential sites that may have the information sought. Consumer agents generally have limited lifetimes, after which they cease to search for information.

Producer agents represent documents similarly to the way in which consumers represent queries. In the case of the ACORN system, the Dublin Core metadata set (which includes description, type, and subject keywords among other attributes) is used to describe a document. InfoSleuth uses KQML and a built-in taxonomy to describe the contents of a document. InfoSleuth also uses another type of agent, the ontology agent, to map terms into a common language for querying between agents. Just as consumers start with a list of potential sites that may have relevant information, producers act as active publishers of information, distributing it to a starting list of readers who may find it of interest. Like consumers, producers can have limited lifetimes. In most cases, the information is archived before the agent expires so that it is still available in a central repository such as a document warehouse. Some multiagent systems perpetuate the life of agents if the information is useful to a large number of consumers, and in some cases, the agent is duplicated to make its information more widely available.

An agent interchange, called a café in ACORN and a broker in InfoSleuth, match consumers and producers. When a consumer finds a producer agent that may be of interest to the consumer's reader, the producer is directed to send the information to the consumer's home server. At the home server, a gatekeeper agent may be used to determine if the information is still relevant to the reader and, if so, add it to the reader's collection of resources.

The use of agents in commercial environments is still limited but we can expect their use to grow. They provide a more flexible architecture that can exploit the processing resources of a distributed network of computers and thus avoid resource bottlenecks on single servers.

Text Analysis Services

Crawlers, agents, and other document retrieval mechanisms will provide the grist for the text analysis mill, which is implemented on text analysis servers. As we have noted in previously, while we make a logical distinction between text analysis tasks, some of them can—and often are—performed in tandem on the same server. Not all document warehouses will need all text analysis services, but the core operations are:

- Character set conversion
- Format conversion
- Machine translation
- Full-text indexing
- Thematic indexing
- Feature extraction
- Summarizing
- Clustering
- Question answering

Character set conversion and format conversion are really preprocessing steps that provide the document in a form suitable for the chosen text analysis tools. Machine translation is a particularly challenging area, and we will leave that for Chapter 8, when we will examine in detail the various options and tradeoffs with machine translation and machine-aided translation techniques. Full-text and thematic indexing and feature extraction tend to overlap but they provide the basis for the most common representation scheme used in information retrieval—the vector model. Once these three operations are performed, we will have an efficient representation of a document for the warehouse. In addition to indexing by the main features, it is often useful to maintain in the warehouse abstracts or summaries of documents. Once documents are represented in the warehouse, we can group them together with clustering techniques. The last operation, question answering, is akin to ad hoc querying in data warehousing.

With these operations in mind, we need to understand key architectural issues such as:

- Given a set of text analysis tools, what preprocessing must be performed?
- What services are bundled together, for example character set conversion and format conversions?
- Given the business intelligence requirements of the project, what text analysis is required and how soon after loading does it need to be done?

Some text analysis tools have preprocessing functions built in directly. Oracle's interMedia Text and AltaVista's Discovery tool can process a wide range of document types, while others require plaintext files.

Some services are bundled together. For example, some tools will generate a full-text index while a document is loaded, and the loading process itself may entail a character set conversion and a format conversion. The benefit of a bundled approach is that there are fewer tasks for the warehouse manager to attend to, but this comes at the expense of not being able to partition the workload by task. In general, only core operations, such as conversions and full-text indexing, are bundled. Optional text analysis tasks and the more computationally expensive operations, such as summarization and clustering, can i 78tes, nally a bun78 02tsteringask8rch ad7(ii operatiore a distributed document warehouse over the Internet, is one approach. Again, there are advantages and disadvantages that need examination.

Database Options

When we think of databases, we often think of relational database management systems (RDBMSs) such as DB2, Oracle, SQL Server, Informix, and Sybase. These certainly dominate the database market, and the data warehousing market in particular, but there are other options. We will consider the three most readily available options for database storage:

- Object databases
- Textbases
- Relational databases

Object databases offer a hierarchical alternative, while textbases treat the document and its attributes as the basic data types. Textbases have evolved to accommodate the needs of text management systems that, at least in the past, have not been well supported in other models. Relational database support for text centers on the basic entity and attribute model so familiar in systems design. Our concern here is not to delve into the details of each database . Our bedatgys that is well(documeneed lsew her)(, but os wigh-)TjT -0..010 Tw(th p

,oer aregculstoyT

document andouseiut osctreahe pecialszeedrvestions in ths fome o morhe pecific-

,earnsinsT

,oerpwseducat stoyT anouoncumentrewse stoyT

documenns

,e pep-

documennsaloinnservecat

Second, object-oriented databases have long provided support for binary objects. Since documents in the warehouse still need to maintain formatting and rendering information, the most efficient method is to store that data along with the document itself, and binary objects are the best way to do that.

Another advantage of object-oriented databases, at least in some cases, relates to the Document Object Model (DOM). DOM is an object model for documents that provides an object-oriented API for managing XML documents. This language-independent descendent of HTML is a natural fit with object-oriented databases.

Textbases

Textbases, such as Lotus Notes and Domino, provide a platform for developing document-centric applications. These applications tend to support a range of information retrieval options, such as complex searches, as well as document-oriented design features, such as folders and electronic file cabinets. The advantages of textbases are their native support for text operations and their ease of use.

Support for text operations often includes more than full-text searching and indexing. Some tools provide for document hierarchies, allowing users to track threads, such as Note's main document, response document, and response-to-response document. Navigation tools for exploring the textbase are also common and in some cases customizable, adding to the tool's ease of use.

Relational Databases

The relational database is the king of the DBMS mountain. With a proven track record for scalability and flexibility in design, RDBMSs are the best choice for document warehousing for several reasons.

First, a broad range of designs can be supported, from generic data models to application-specific models, such as dimensional models in data warehousing. David Hay's *Data Model Patterns* (Hay 1995 demonstrates the flexibility of generic data models in several areas, including document management. Ralph Kimball's *Data Warehouse Toolkit* (Kimball, 1996) and Kimball et al.'s *Data Warehouse Lifecycle Toolkit* (Kimball 1998) develop commonly used techniques for dimensional models that provide for efficient and easily navigated data warehouses.

A second benefit of using an RDBMS is that we can support multiple types of text indexing structures, such as full-text indexes, thematic indexes, and multidimensional taxonomies, using the same basic RDBMS features.

Third, and perhaps most importantly, with text mining and document warehousing, we are not just interested in treating documents as a monolithic unit or a hierarchy of subdocuments. We want access to the semantic content of the document, and to get this we must have a way to store and access the key attributes about the document such as its:

- Main topics
- Names of companies, government agencies, or individuals mentioned
- Relationships between those organizations and individuals
- Other metadata

All of these items easily fit the tabular model of relational databases. In addition, the demands of data warehouses have pushed relational database vendors to develop systems that can manage large (terabyte-plus-sized) databases. Now, we have the benefits of a natural fitting model with the scalability and performance needed for large document warehouse projects.

The Metadata Repository and Document Data Model

In addition to storing documents, which could fit into any of the three database models just discussed, we need to store information that describes those documents and that has been extracted from those documents. This information constitutes the metadata about the document and comes in four types:

- Document content metadata
- Search and retrieval metadata
- Text mining metadata
- Storage metadata

Document content metadata generally follows one of the document metadata standards such as the Dublin Core or the Text Encoding Initiative. Search and retrieval metadata contains information about when documents were added and how they were found. Text mining metadata is basically the output from text mining operations. Storage metadata is used for managing the contents of the document warehouse.

Document Content Metadata

While a document warehouse designer can arbitrarily define the document content metadata for the warehouse, it is strongly suggested that you use one of the widely adopted standards as a basis. The main reason for this is that there is a growing awareness for the need for metadata on the Web. Organizations are adding more metadata to the content of their Web sites to ease *their own* content management problems. As intranets grow within organizations and sometimes merge, they become more difficult to navigate and searching with standard tools still brings on the problems of poor precision and recall. As a result, there is a heightened interest in using metadata to improve the value of intranets and the Web in general. The Dublin Core metadata standard was developed with Internet resources in mind and is the best option for most document warehouses. If your warehouse will be integrated with other systems that depend upon other metadata standards, such as geographic information systems, then by all means use those standards. Regardless of the standard used, when collecting documents from sites that use the same standard as you do, the metadata can be easily extracted and categorized within the document warehouse.

Here is a basic list of attributes to start with when designing content metadata:

- Creator
- Subject
- Title
- Description
- Publisher
- Contributor
- Published Date
- Revised Date
- Type
- Format
- Language
- Rights

Search and Retrieval Metadata

As we will see in Chapter 8, the process of managing the search and retrieval process is dependent upon metadata about what types of documents to look for. Unlike document content metadata, which describes the features of a particular document, the search and retrieval metadata describes:

- How the document came to be in the document warehouse
- What kind of source it came from
- How to manage multiple versions

How documents come into the warehouse is defined by source metadata that includes URLs for online sources or file system pathname for local files, frequency of searches, regular expressions specifying search criteria, and crawler- or agent-specific information that controls the search processes. For example, some crawlers allow users to specify a time interval between requests to servers. This is especially useful when searching sites that could easily be taxed by rapid page hits.

Not all sources are the same, and users of the warehouse will need information about where documents came from. The most important attribute about a source is its quality. For document-centric applications, this can be distilled down to timeliness and veracity. News sources are in constant competition to deliver breaking news, so they are generally very timely sources. On the other hand, some checkout counter tabloids may be timely, but the accuracy of their content is questionable. For complex issues, such as those warranting government investigations, such as airline crashes, accuracy is more important than timeliness when measuring quality. Timeliness and veracity are the two core measures for quality but you may find the need for others, depending upon your applications.

Another issue that should be dealt with during the search and retrieval process is versioning. Frequently, one will repeatedly find the same document at a source. Checking the timestamp of the source is the simplest way to determine if this is a new version and should be downloaded. Operating systems that support versioning will provide other means for detecting changes in versions as well. In some cases, as with draft documents, contracts, position papers, or other evolving texts, you may want to store all versions of the document instead of replacing the existing version in the warehouse. (This raises a distinction between data warehouses, where data is rarely, if ever deleted, and document warehouses, where some texts do not warrant perpetual storage. There will be more on this topic in Chapter 9.)

Here is the basic list of search-and-retrieval metadata to begin with:

- Source Type
- URL Pattern
- FilePath
- Depth
- Span Site Indicator
- Number of Tries
- Time out
- Wait between Retrievals
- HTTP UserName

- HTTP Password
- Proxy UserName
- Proxy Password
- Reject List
- Include Directories
- Exclude Directories
- Search Engine

With this set of search and retrieval metadata one can precisely control the document collection process.

Text Mining Metadata

The content of text mining metadata is driven by the specific needs of the document warehouse users. The most basic elements of text mining that metadata will include document:

- Features, such as keywords and topics
- Summaries
- Relations

Features identify the main points of a document. Depending upon the tools used, features may be automatically stored in a database and managed behind the scenes, much like indexes in a relational database. In cases where they must be dealt with explicitly, keywords can be stored as attributes of a document along with a measure of their frequency within the document. Topics similarly are stored along with a measure of the importance of the topic to the overall document. In most cases, documents will have multiple keywords and topics.

Depending upon the expected use of certain document types, summaries can be generated after the documents are loaded into the warehouse or on an as-needed basis. Since summaries describe the content of a document, they are, by definition, metadata, although metadata generally falls into more of a conventional attribute value form.

Relations describe the relationships between persons, places, and things in a document. A financial news story that includes text such as:

> Mary Smith, president and CEO of Alpha Industries, announced the acquisition of Gamma International of Montreal at a press conference today.

Identifies two relations directly, as shown in Table 6.1.

Table 6.1 Relations Describe Roles and Attributes of Entities within a Text

ENTITY1	RELATIONSHIP	ENTITY2
Mary Smith	President and CEO	Alpha Industries
Gamma International	Location	Montreal

Feature extraction programs can identify these relationships as well as data and time expressions, ages of persons, proper names, and others. The fact that a precise relationship can be established allows users to search the metadata of relations with precision rather than having to search text with criteria such as "Mary Smith" AND "president."

Document features should be tied to a taxonomy of features to allow the greatest search and navigation flexibility within the document warehouse. A taxonomy, sometimes called a ontology to distinguish it from navigational taxonomies such as Yahoo!, is a knowledge representation scheme for organizing relationships between terms. Terms relate to each other as either broader terms or narrower terms. Taxonomies can also be represented as thesauri (see Chapter 8), which add support for preferred term relations. A single concept can be expressed in several forms, for example the terms *President*, *President of the United States*, *Chief Executive*, and *Commander in Chief* all refer to the same position. Rather than integrate all of these terms into a taxonomy, a single term is chosen as the preferred term used in the taxonomy. The synonymous terms are mapped to the preferred term during indexing.

Storage Metadata

Storage metadata describes how a document should be handled once it has been retrieved and analyzed. The main types of storage metadata deal with how to represent the document in the warehouse and when to change representations.

Let's describe the possible representation schemes. First, we can store the entire document. That sounds pretty straightforward, but like everything else in the world of IS, things are not as simple as they first appear. Documents may need to be translated and reformatted. Should the original document be kept along with the translated version? How many different translations should be kept? Should only summaries of the translation be kept? Machine translation, while reasonable, is not perfect, as we shall see in Chapter 8. Should we keep a machine-translated version of the document until a human translator reviews and corrects it? Depending upon the specific needs of your text mining applications, additional storage metadata may be required. In general, though, the issues will center on how to handle multiple representations of the same content.

Instead of storing the entire document, we may decide, for reasons of balancing storage with document priorities, that not all documents need to be stored in their entirety. We could store only summaries or just a URL. In the later case, we should specify when the URL should be verified to ensure that it still exists.

In some cases, the timeliness of some documents may require that they be stored in the document warehouse at first but after a period of time can be effectively archived. One approach in this situation is to store the entire document at first and then store only the summary and eventually just a URL or other reference to its location. As mentioned previously, this process of reducing the document representation within the warehouse is called pruning and can be an effective strategy for balancing the need for controlled access with the need to manage storage.

Here is the basic list of storage metadata to begin with:

- Store Entire Document Indicator
- Store Summary Indicator
- Store URL Indicator
- Store Pathname Indicator
- Prune Full Document
- Prune Full Document after
- Prune Summary Indictor
- Prune Summary after
- Keep Full Translations
- Keep Summarized Translations
- Translation Review Required

Storage metadata such as this is associated with a set of documents, such as particular document types, documents from a specific source, or those found by using particular search criteria.

Document Data Model

With the discussion of document metadata, we have made a transition from the physical architecture of the document warehouse to the logical design considerations. The document data model is at the heart of the logical model of the warehouse. As Figure 6.9 depicts, the document warehouse logical model has similarities to the star schema architecture used so often in data warehouses.

Content, source, search, and storage metadata correspond to dimensions in a dimensional model. Although specific content metadata will often relate to only

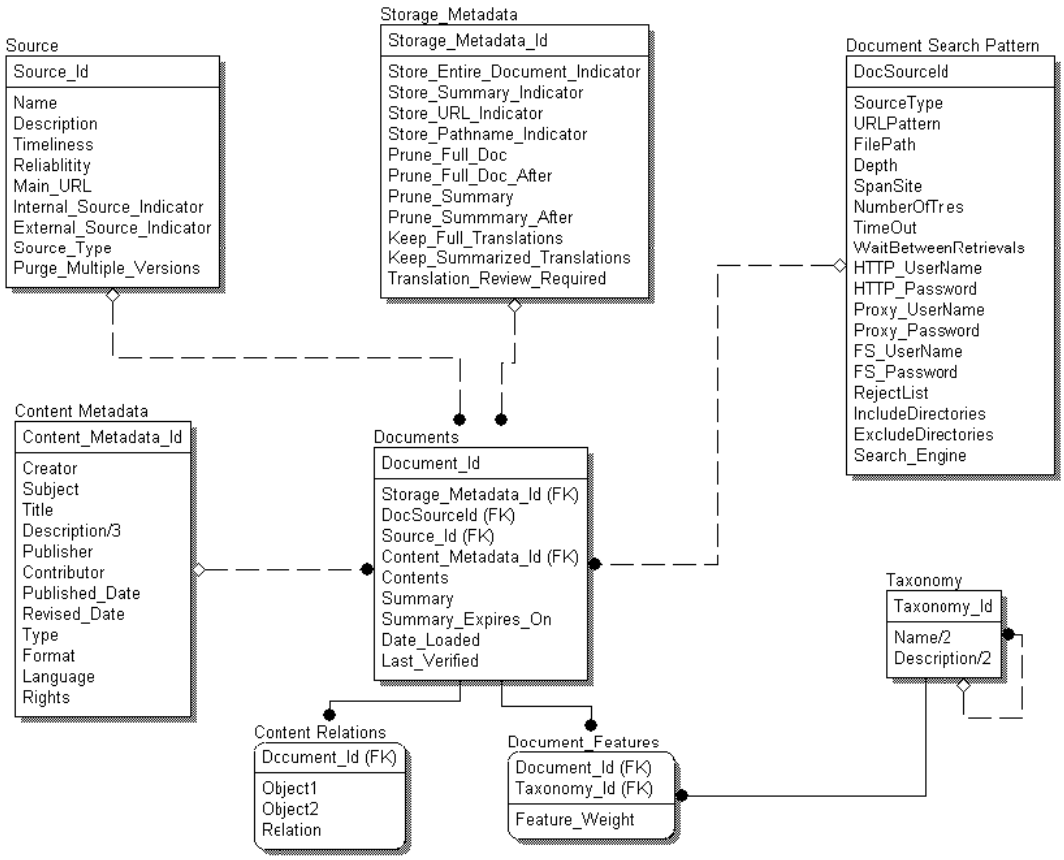


Figure 6.9 A basic model for the document warehouse.

a single document, when multiple versions are stored, it may relate to several documents. Documents have a many-to-one relationship with content relations and document features. Features ideally correspond to the preferred terms used in a thesaurus or taxonomy.

The text of a document is stored, along with optional summary, in the documents table. Versions are tracked as well at this level. Since documents may be pruned or deleted, expiration dates are also tracked. Finally, if only a URL is kept in the documents table, a record is kept of the last time the URL was verified. For a detailed listing of this model, see Appendix B and the companion Web site.

User Profiles and End-User Support

While many users have one-time questions that can be answered by ad hoc queries to a data warehouse or a by a question-answering system in a document warehouse or over the Internet, these same users often have long-term interests as well. End-user profiles are commonly used to represent the long-term interests of users. By keeping a long-term record of a person's interests, we can utilize information-filtering techniques, such as agents, to improve the quality of information provided to our users. The key issues in user profiles are:

- How should we represent user's interests?
- How should we create a user profile?
- How should we maintain a user profile?

Finding a way to represent a user's interest is perhaps the most important issue, for the profile must be capable of expressing the domains of interests while eliminating topics that are irrelevant. Creating a user profile depends upon the representation scheme utilized and the method for gathering initial information. Finally, maintaining a profile over time requires us to recognize when interests change and to adjust the profile in such a way as to cover new interests and eliminate old interests without adding too many new topics or removing others still of genuine interest to the user. In many ways, these three issues boil down to the classic problem of information retrieval, how to balance precision (getting only documents of interest) with recall (not missing documents of interest).

The most common way of representing a user's interests is to use a list of terms. This is similar to the list of terms that is used to represent documents in the vector model of information retrieval. Generally, terms are weighted so that those that are more important have larger weights than those of less importance. For example, someone interested in document warehousing may have a list such as the one in Table 6.2.

Table 6.2 Weights Are Used to Indicate the Relative Importance of Different Topics to Users

TERM	WEIGHT
Agents	0.72
Data Warehouse	0.74
Document Warehouse	0.95
Hypertext	0.48
Information Retrieval	0.60
Latent Semantic Indexing	0.57
Morphology	0.63
Text Categorization	0.82
Text Mining	0.98

Another approach to representing user interests is through filtering rules. These generally take the form of "IF <condition> THEN RELEVANCY = <value>". The condition specifies a known attribute about the user, such as the department she works for or her position within the organization, or attributes about documents, such as the extracted features or a metadata item (for example, the author's name):

```
IF features = { hypertext and information retrieval and XML }
THEN RELEVANCY = 0.67;
```

In both representation schemes, the same key words or features that represent documents are used in user profiles, so matching documents with interested users becomes a matter of matching common terms and weights. These representation schemes can be used in a number of different types of user profiles. Researchers in user profiling (Kuflik and Shoval 2000) have identified six methods for user profile creation:

- User-created profiles: With this method a user specifies key areas of interests and optionally provides weights for each domain. Customized news services, such as myCNN.com, use this approach.
- System-created profiles by automatic indexing: Users specify a group of documents that they deem interesting, and the system identifies the most frequent and meaningful terms to use as the basis for the profile.
- System plus user-created profiles: In this method, an initial profile is created by automatic indexing, and then the user modifies the results.
- System-created profile based on learning by neural networks: With this method, a neural network is trained using a set of documents judged relevant by a user. The network is then used with other documents to determine their relevance relative to the training set of documents.
- User profile inherited from a user stereotype: This method assumes that an administrator has defined templates of common interests; for example, for financial analysts, quality control engineers, competitive intelligence analysts, and so on.
- Rule-based filtering: This approach builds a set of rules on the basis of a set of standard questions posed to a user about his or her information needs. This technique may be combined with user stereotypes to provide a starting set of rules that is then modified according to user specifics.

Again, for our purposes of answering questions about the architecture of the document warehouse, we need to decide what approach is the best for a particular document warehouse. User-created profiles, system-created profiles by automatic indexing and system plus user-created profiles are the easiest to implement. Full-blown rule-based filtering requires a rule processing engine that can resolve conflicts between rules and prioritize rules. The additional overhead is probably not worth the improvement, if any, in precision and recall over other methods. Keeping two metaobjectives in mind—that is, the document warehouse must be easy to use and it must be fast—we should opt for weighted term vector models for representing user profiles. They will be easier to utilize in a production environment.

End-User Profiles

As Kuflik and Shoval have shown, there are several ways to represent user interests. At this point, we shall briefly turn to the basics of modeling user interests.

User-Created Profiles

User-created profiles are the easiest of the several techniques mentioned previously. User-created profiles often begin with simple checklists based upon taxonomy entries as shown in the following list.

A SIMPLE CHECKLIST TAXONOMY FOR CREATING USER PROFILES

- ☐ Business
  - ☐ Finance
  - ☐ Marketing
  - ☐ Accounting
  - ☐ Sales
- ☐ Science
  - ☐ Physical Science
  - ☐ Astronomy
  - ☐ Chemistry
  - ☐ Meteorology
  - ☐ Physics
- ☐ Biological Sciences
  - ☐ Anatomy
  - ☐ Genetics
  - ☐ Marine Biology
  - ☐ Physiology
  - ☐ Zoology
- ☐ Social Sciences
  - ☐ Politics

- International Relations
- Federal Government
- Law and Judiciary
- Regional Politics
  - Economics
  - Macroeconomics
  - Fiscal Policy
  - Monetary Policy
  - Microeconomics
- Stock Markets
  - Bonds and Debt Instruments
  - Sports
  - Baseball
  - Golf
  - Football

As the above example illustrates, user-created profiles can be based upon a hierarchical structure and thus tied to a taxonomy. The data model required to support this type of profile is extremely simple and is shown in Figure 6.10.

Rule-Based Profiles

The list-of-terms model for user interests can be captured in a data model similar to the one depicted in Figure 6.10. Filtering rules based upon conjunctive and disjunctive Boolean conditions can also be modeled with minor modifications. For example, the following rule provides for individual weighting of each term.

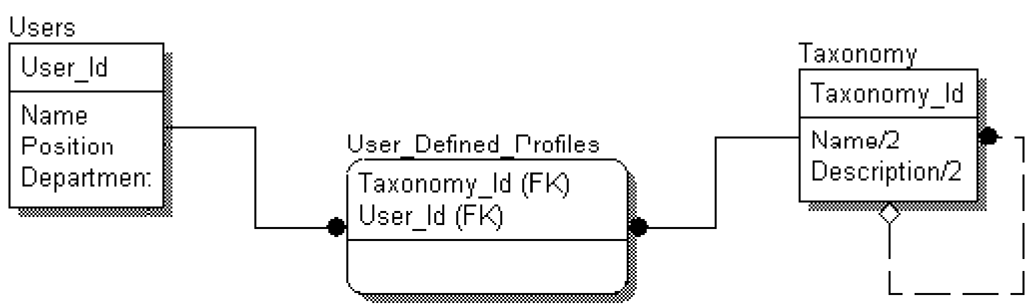


Figure 6.10 Modeling a user-created profile requires as few as three tables.

```
IF (hypertext > 0.68) AND
   (information retrieval > 0.50) AND
   (XML > 0.3)
THEN RELEVANCY  = 1;
```

This row, would require three rows in the User_Interests table. The taxonomy_id in each row would correspond to either hypertext, information retrieval, or XML. The minimum_weight is simply the weight specified in the Boolean condition. Interest_Group_Id is a unique identifier allowing us to group the three conditions together. Disjunctive conditions can be similarly be represented by using additional interest groups.

This type of rule representation does, however, have its limits. First, it assumes that all rules will be composed of simply a set of minimal weights for particular features. Second, disjunctive rules must be represented as distinct rules, so that a rule of the form

```
IF (hypertext > 0.68) OR
   ((information retrieval > 0.50) AND
    (XML > 0.3))
THEN RELEVANCY = 1;
```

will require two separate interest group sets, one with hypertext and information retrieval, and one with hypertext and XML. In spite of these limits, if a rule-based approach is considered, the simpler it is, the better.

User interests are not always linked to text. We saw in Chapter 1 that data warehouses cannot meet all the business intelligence needs of decision support system users. Document warehouses cannot meet all their needs either, as the importance of data warehouses can attest.

Data Warehouse and Data Mart Integration

Document warehouses will frequently coexist with data warehouses. As we saw in Chapter 3, analysis in a data warehouse environment can lead to questions that cannot be answered by looking at numbers. This can lead users to the document warehouse in search of text on a particular subject, which may in turn raise questions or theories that must be checked against the data warehouse. As Figure 6.11 shows, business intelligence operations often move between the realms of text and numbers.

As we address design issues relating to the integration of data warehouses and document warehouses, we will work on three topics:

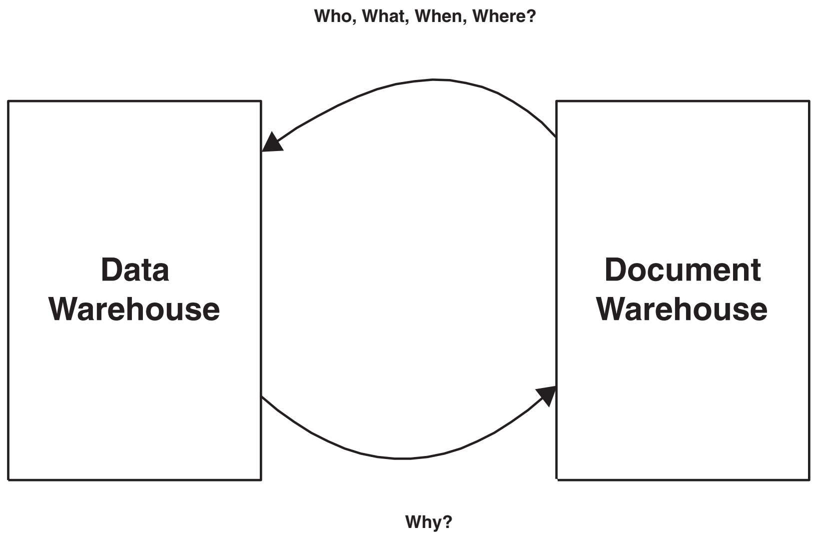


Figure 6.11 Analysts need both text and numeric data to form a complete picture of a business situation.

- Linking numbers and text
- Integration heuristics
- Limits of automated integration

Linking numbers and text is our primary goal. To meet this objective, we will consider how metadata can be used to serve integration and discuss some heuristics, or rules of thumb, for exploiting the features of dimensional models to link the two types of warehouses. Finally, we will examine the limits of these automated techniques.

Linking Numbers and Text

Data warehouses are filled with measurements about sales, revenues, production quotas, budgets, and other business processes and entities. Dimensional warehouses allow us to quickly and easily target measures to particular aggregated areas of interest, such as the sales figures of kitchen appliances in the northeast region last summer or particular measures such as the revenues from the sale of 19" Sony televisions in store number 874 last week. Now, it is not likely that we will find detailed documents about the sale of 19" Sony televisions in store number 874 last week or at any other time. We will, however, find documents about the product type, such as electronics and the region of the store. And this leads to a general principal about linking numbers and text: The precision of numbers will outstrip the precision of text.

Table 6.3 Basic Attributes of a Product Dimension in a Data Warehouse

ATTRIBUTES	EXAMPLE VALUES
Product_Id	378456
Product Category	Consumer Electronics
Product Class	Televisions
Product Type	19" HDTV
Manufacturer	Sony
SKU	338764531
…and others	

To adequately link numbers and text, we must generalize from specific items, such as 19" Sony televisions or store 874, to higher levels within data warehouse dimensions.

For example, a product dimension in a dimensional data warehouse may have the attributes shown in Table 6.3.

For the purposes of finding related documents, the keywords and topics are *consumer electronics*, *televisions*, and *HDTV*. Combining these with similar types of attributes from other dimensions, such as geographic regions, can provide the starting point for linking the document warehouse and the data warehouse.

Integration Heuristics

In data warehousing operations, we are accustomed to dealing with algorithms—that is, a sequence of operations that produce a well-defined result. The wonderful thing about algorithms is that, once implemented correctly, they always produce the correct result. Heuristics, on the other hand, are general rules of thumb that work in many cases, but not all and so, on occasion, they produce incorrect or poor results. When it comes to integrating data warehouses and document warehouse, we are limited to heuristics. Three generally applicable rules for generating keyword descriptors to link the two warehouses are:

- Ignore numeric measures from descriptions.
- Weight keywords proportionally to their depth in the hierarchy.
- Make sure words used in descriptions are in the taxonomy.

Dimensions in a data warehouse are loaded with descriptions. Some are very detailed and others quite broad, depending upon the level of the dimension hierarchy that is being described. Frequently the most detailed descriptors contain numeric measures, such as 19" television, that can be ignored since they do not typically lead to significant improvements in precision or recall. Note that this does not include model numbers, such as a GM Suburban 1500 LT, which can be used for more precise searching in the document warehouse.

When searching for related documents, weighting precise descriptors more heavily than general labels, such as consumer electronics, will also improve precision and recall. Since the general labels can apply to so many different documents, especially in thematically indexed warehouses, weighting less frequently used terms more heavily will rank documents with these terms higher than if the weightings had not been used.

Finally, because terms used in dimension descriptions are the basis for keyword or topical searches in the warehouse a common vocabulary will improve integration. Taxonomies are frequently used in text mining tools to support topical searches and browsing. Terms used in dimensional descriptions should be included in the document warehouse taxonomies as well.

Conclusions

Designing a document warehouse entails a range of decisions—where to get documents, how to get them, what to do with them, how to manage what you get, how to help users get what you have to offer. The key decision points to keep in mind when designing the warehouse are:

- Where do we find the documents that meet the users' requirements?
- How should the documents be extracted?
- What transformation and text mining operations should be applied?
- How should different classes of documents be handled in terms of storage and analysis?
- How do we provide support for long-term user interests?
- How do we integrate the document warehouse with the data warehouse?

Some of these issues can be addressed relatively quickly, such as how to extract documents once their source has been identified. Others, such as integrating data and document warehouses, will likely evolve over time. Document warehousing and text mining are relatively young disciplines in organizational settings, and in time more heuristics and other design principals will emerge. In the meantime, addressing the six main design issues described above will put the document warehouse on a firm foundation for growth.